

PROJEKT

TEORIA I METODY OPTYMALIZACJI

---

# Harmony Search

---

*Autor:*

Artur GASIŃSKI, 218685

Bartosz LENARTOWICZ, 218518

*Termin:* wt 9:15-11:00

*Prowadzący:*

Dr inż. Ewa SZLACHCIC

28 maja 2018

# Spis treści

<b>1</b>	<b>Sformułowanie zadania optymalizacji</b>	<b>2</b>
<b>2</b>	<b>Szczegółowe omówienie algorytmu optymalizacji</b>	<b>2</b>
<b>3</b>	<b>Informacje o programie</b>	<b>3</b>
3.1	Środowisko programistyczne, zastosowane biblioteki i gotowe moduły . . . . .	3
3.2	Parametry . . . . .	3
3.3	Wykres . . . . .	3
3.4	Tabela rozwiązań . . . . .	4
<b>4</b>	<b>Przykłady rozwiązań oraz testy</b>	<b>5</b>
4.1	Funkcja czterech minimów . . . . .	5
4.1.1	Wyznaczenie wszystkich minimów . . . . .	5
4.2	Funkcja testowa Geema . . . . .	8
4.2.1	Zmiana zakresu dla argumentów . . . . .	8
4.3	Funkcja Rastrigina dla $n = 3$ . . . . .	10
4.4	Funkcja Himmelblau . . . . .	10
4.5	Funkcja Goldsteina-Price'a z czterema minimami lokalnymi . . . . .	10
4.6	Dwuwymiarowa funkcja sinusoidalna . . . . .	10
4.7	Dwuwymiarowa funkcja sinusoidalna z eksponentą . . . . .	15
4.8	Podsumowanie . . . . .	15
<b>5</b>	<b>Wnioski końcowe</b>	<b>17</b>
	<b>Literatura</b>	<b>18</b>

# 1 Sformułowanie zadania optymalizacji

Większość metod optymalizacji procesów produkcyjnych czy logistycznych sprowadza się do przedstawienia zależności pomiędzy procesami za pomocą określonej funkcji celu  $f(x)$ . Pod postacią zmiennych  $x$  tego równania ukazane są parametry procesów, którymi można manipulować. By właściwie zoptymalizować proces, należy znaleźć minimum bądź maksimum (najlepiej globalne) funkcji  $f(x)=f^*(x^*)$ . Obliczanie minimalnej bądź maksymalnej wartości  $f^*$ , zależnej od wielu zmiennych  $x$  w pożądanym czasie jest problematyczne. Istnieją algorytmy specjalizujące się w rozwiązywaniu zadań tego typu. Niniejsza praca została napisana by przedstawić jeden z nich – Harmony Search. Został on przedstawiony w 2001 przez Zong Woo Geema, Joong Hoon Kima oraz G.V. Loganathana w pracy „A New Heuristic Optimization Algorithm: Harmony Search” [1].

## 2 Szczegółowe omówienie algorytmu optymalizacji

Inspiracją dla algorytmu było szukanie przez muzyków jazzowych najlepszych harmonii dźwięków podczas improwizacji. Jego matematyczny odpowiednik umożliwia optymalizację funkcji  $f(x)$  wielu zmiennych.

Algorytm do działania wymaga zainicjowania  $m = 4 + 2 * n$  wartości początkowych. Jedynym kryterium stopu algorytmu jest liczba iteracji  $L$ . Głównym elementem algorytmu jest pamięć harmonii HM (*ang. harmony memory*). Jest to zbiór zapamiętanych rozwiązań funkcji  $f(x)$  wraz z wartościami argumentów  $x$ , dla których została obliczona funkcja. Schemat tablicy HM umieszczony jest na rysunku 1. Wielkość pamięci  $HMS$  deklarowana jest z góry przez użytkownika i podczas kolejnych iteracji nie zmienia rozmiaru. Wymagane jest również by na początku działania określić przedział wartości  $[a, b]$  dla każdej zmiennej  $x_1 = [a_1, b_1], \dots, x_n = [a_n, b_n]$ , z którego będą wyszukiwane rozwiązania dlatego należy zadeklarować  $2 * n$  wartości gdzie  $n$  jest ilością zmiennych równania. Dodatkowo wymagane jest określenie dwóch prawdopodobieństw. Pierwsze  $HMCR$  (*ang. harmony memory consideration ratio*) określa współczynnik doboru tonu z pamięci, drugie  $PAR$  (*ang. pitch adjustment ratio*) nazywane jest współczynnikiem dostosowania tonu. Oczywiście jest, że przedział wartości prawdopodobieństw określany jest na  $< 0, 1 >$ .

Algorytm działa następująco: w pierw program wypełnia tablicę pamięci harmonii HM początkowymi wartościami. Każda wartość  $x$  losowana jest z swojego ograniczonego przedziału  $[a, b]$ . Gdy tablica zostanie zapełniona rozpoczyna się część iteracyjna algorytmu. Polega na wyszukiwaniu kolejnego rozwiązania poprzez wyznaczanie kolejnych rozwiązań wedle trzech reguł. Pierwsza mówi, że z prawdopodobieństwem  $P_1 = HMCR$  wartość nowej zmiennej  $x_n$  wybierana jest z pośród zmiennych zapamiętanych w tablicy HM. Druga, że wartością prawdopodobieństwa  $P_2 = (1 - HMCR)$  wartość zmiennej losowana jest z całego przedziału  $[a, b]$ , a zasada trzecia, że zmienna ustalona w regule pierwszej modyfikowana jest z prawdopodobieństwem  $P_3 = (HMCR * PAR)$ . W praktyce reguła trzecia sprowadza się do dodania do wylosowanej zmiennej  $x$  z reguły pierwszej liczby z przedziału  $< -c, c >$ . Następnie gdy zostaną wylosowane wszystkie wartości  $x_n$  obliczana jest dla nich wartość funkcji  $f(x)$ . Jest ona porównywana z wartościami umieszczonymi w tablicy HM. Jeżeli jest mniejsza niż największa wartość z tablicy, zamienia się ją z największą wartością (najgorszym rozwiązaniem). Na tym kończy się iteracja. Następnie wykonuje się kolejne iteracje do osiągnięcia maksymalnej ilości iteracji  $L$ . Polski opis algorytmu można również znaleźć np. w pracy [2].

$x_{11}$	$\dots$	$x_{n1}$	$= f_1$
$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{1HMS}$	$\dots$	$x_{nHMS}$	$= f_{HMS}$

Rysunek 1: Schemat tabeli *Harmony Memory*

### 3 Informacje o programie

Program do obliczania minimum funkcji na podstawie algorytmu Harmony Search został napisany w języku Java. Technologią dostarczającą GUI dla użytkownika była *JavaFx* umożliwiająca tworzenie aplikacji okienkowych. Dodatkowo by ułatwić zarządzanie bibliotekami wewnątrz projektu użyto wsparcia ze strony *Maven*. Program korzysta z zewnętrznej biblioteki *mXparser* do parsowania łańcuchów znakowych na funkcje i do obliczania ich wartości w określonych punktach [3]. Wykres prezentujący najlepsze rozwiązanie zaznaczone na wykroju z przestrzeni rozwiązań funkcji został stworzony za pomocą biblioteki *jzy3D*.

#### 3.1 Środowisko programistyczne, zastosowane biblioteki i gotowe moduły

By ułatwić wprowadzanie funkcji do programu oraz bezproblemowo odczytywać wyniki obliczone przez program zostało stworzone GUI programu. Na rysunkach 2, 3 i 4 przedstawiono etapy działania programu. Poszczególne elementy GUI podczas działania programu nie zmieniają swojego położenia. Jedynie pustych w miejscach zostają wyświetlone ich parametry, dlatego by omówić elementy posłużono się rysunkiem 4, gdzie widnieje obraz programu po obliczeniach. Główne okno programu zostało podzielone pionowo na trzy części. Pierwsza część od lewej przeznaczona jest do wpisania funkcji i jej parametrów, środkowa prezentuje wykres, a część prawa w tabelce wyniki obliczeń programu. Wszystkie części zostaną szczegółowo opisane poniżej.

#### 3.2 Parametry

W części parametrów można dostrzec, że zaraz pod paskiem edycji znajduje miejsce do wpisania funkcji, której minimum należy wyznaczyć. Okno umożliwia również wybór takiej funkcji z wcześniej zadeklarowanych w programie. Dokonuje się tego, klikając w rozwijane menu. Zostało to przedstawione na rysunku 2. Po wybraniu funkcji automatycznie zostaną domyślnie uzupełnione wszystkie parametry obliczeń. Oczywiście możliwa jest dostosowanie algorytmu do własnych potrzeb. Poniżej okna funkcji znajduje się przycisk „Resetuj wartości parametrów” umożliwiający, jak mówi nazwa przywrócenie domyślnych wartości parametrów. Poniżej przycisku zostały umieszczone po sobie cztery główne parametry algorytmu *Harmony Search*.

Pierwszy z parametrów **HMS** (*ang. harmony memory size*) reguluje ilość zapamiętywanych rozwiązań podczas wyszukiwania. Domyślnie rozmiar jest zadeklarowany na 20. Poniżej znajduje się okno o nazwie **iteracje**. Domyślnie jest zadeklarowana maksymalna wartość iteracji równa 10000. W każdej iteracji programu algorytm wyszukuje nowe rozwiązanie i sprawdza je z tymi zapisanymi w **HMS**. Im większa liczba iteracji, tym algorytm powinien obliczać dokładniejsze rozwiązanie, kosztem wydłużonego czasu obliczeń. Następnie pod polem **iteracje** znajduje się pole **HMCR** (*ang. harmony memory consideration ratio*). Jest to prawdopodobieństwo wyboru rozwiązania z pamięci (z zakresu  $(0, 1)$ ). Im **HMCR** jest większe tym wyszukiwane wartości  $x$  następnych rozwiązań będą zbliżone do tych istniejących już w tablicy **HMS**. Poniżej znajduje się drugi parametr prawdopodobieństwa o nazwie **PAR** (*ang. pitch adjustment ratio*) tłumaczony jako współczynnik dostosowywania tonu. Wybierany jest również z zakresu  $(0, 1]$ .

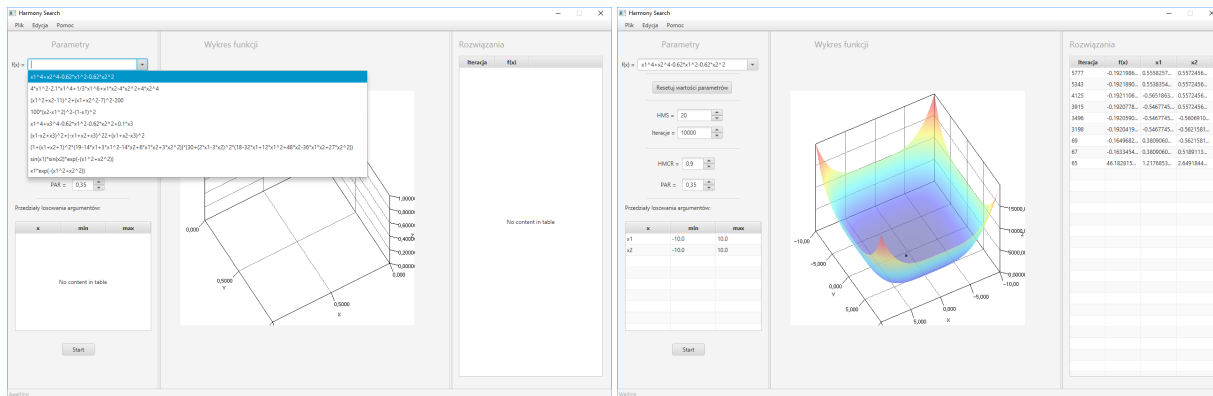
Na samym dole kolumny znajduje się tabela **Przedziały losowania argumentów**. W tabeli deklarowany jest przedziały wszystkich wartości  $x$ , z których zostaną wylosowane początkowe argumenty podczas inicjalizacji pamięci harmonii. Domyślnie parametry dla wszystkich  $x$  brane są z przedziału  $(-10, 10)$ . Algorytm z dużą dokładnością znajdzie najmniejsze minimum lokalne dla funkcji z podanego przedziału. Po wybraniu wszystkich parametrów można przystąpić do uruchomienia algorytmu przyciskiem **Start** znajdującym się na dole kolumny.

#### 3.3 Wykres

Środkowy obszar okna głównego przeznaczony jest do wyświetlania wykresu funkcji. Jak wspomniane zostało na początku rozdziału 3, do rysowania wykresów została użyta zewnętrzna biblioteka *jzy3D*. Umożliwia ona tworzenie wykresów dwu- i trójwymiarowych. Wykresy można obracać w dowolny sposób. Oś  $X$  i  $Y$  wykresu wyskalowana jest do zgodnie z podanymi przedziałami losowania argumentów  $x$ . Dodatkowo wartości zostały pokolorowane w tak, że mniejsze wartości funkcji mają kolory zimniejsze, a wyższe - kolory cieplejsze, analogicznie do tego jak przedstawia się wysokość n.p.m. w kartografii. Dodatkowo czarną kropką zaznaczona została najmniejsza obliczona wartość funkcji.

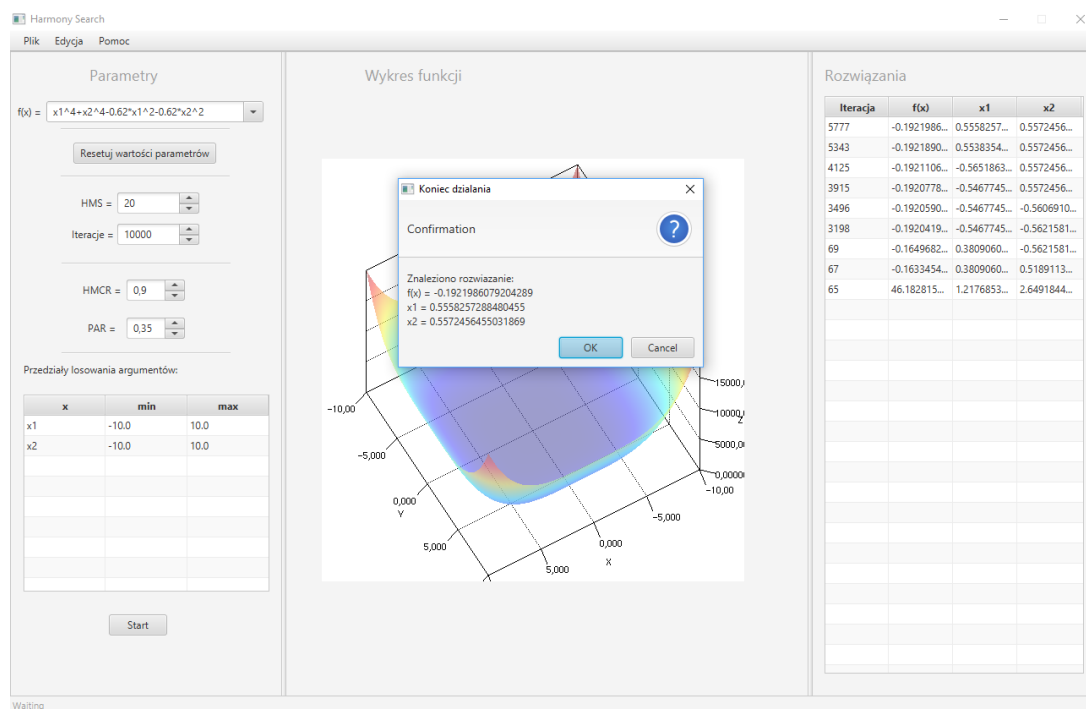
### 3.4 Tabela rozwiązań

Po prawej stronie okna głównego można dostrzec najlepsze rozwiązania wyszukane przez program. Gdy znajdowane jest rozwiązanie lepsze od najlepszego rozwiązania zapamiętanego w HMS dodawane jest ono jako pierwszy wiersz w tabeli. Poprzednie rozwiązania przesuwane są o jedno miejsce niżej. Pozwala to śledzić jaka wartości jest aktualnie największa oraz od razu ukazuje się obok numer iteracji tego rozwiązania. Gdy algorytm długo oblicza najlepszy wynik, okno pozwala śledzić postęp obliczeń. Dodatkowo po wykonaniu wszystkich iteracji zostaje wyświetlony komunikat z najlepszym rozwiązaniem (pierwszym od góry), co przedstawia rysunek 4.



Rysunek 2: Wpisywanie funkcji

Rysunek 3: Wykres z zaznaczonym rozwiązaniem



Rysunek 4: Obliczone rozwiązanie wraz z komunikatem

## 4 Przykłady rozwiązań oraz testy

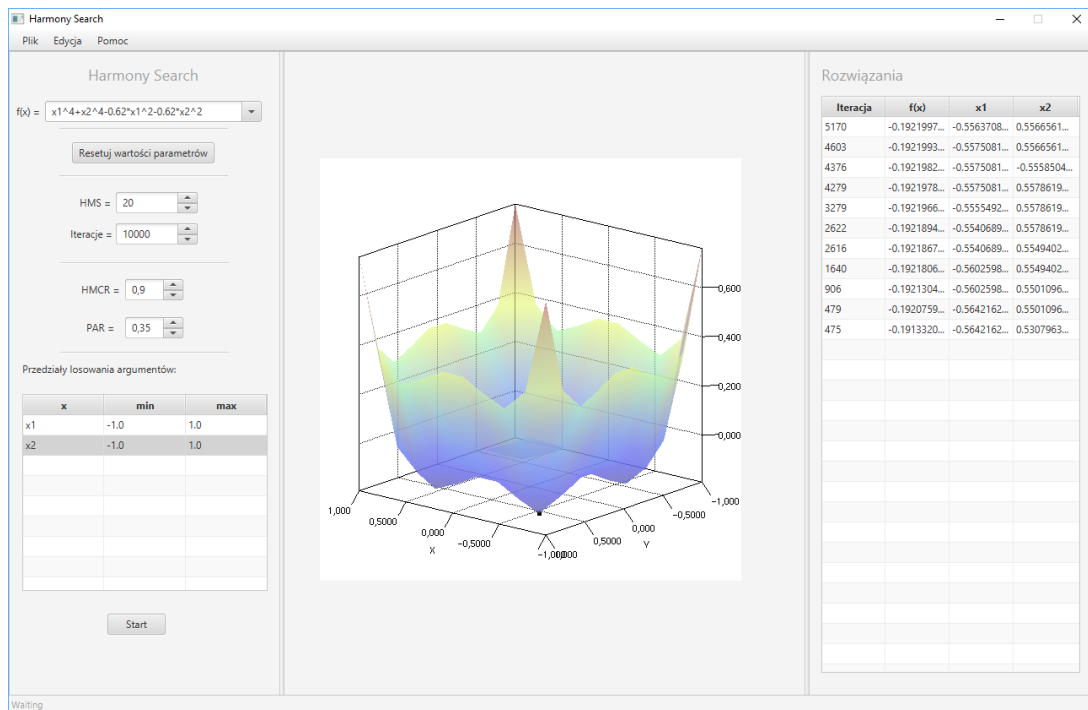
W tej części zostały przedstawione przykłady funkcji wraz z wynikami obliczonymi dzięki przedstawianemu programowi.

### 4.1 Funkcja czterech minimów

Pierwsza zostanie przedstawiona funkcja posiadająca cztery minima lokalne. Wyraża się ona wzorem:

$$f(x_1, x_2) = x_1^4 + x_2^4 - 0.62x_1^2 - 0.62x_2^2.$$

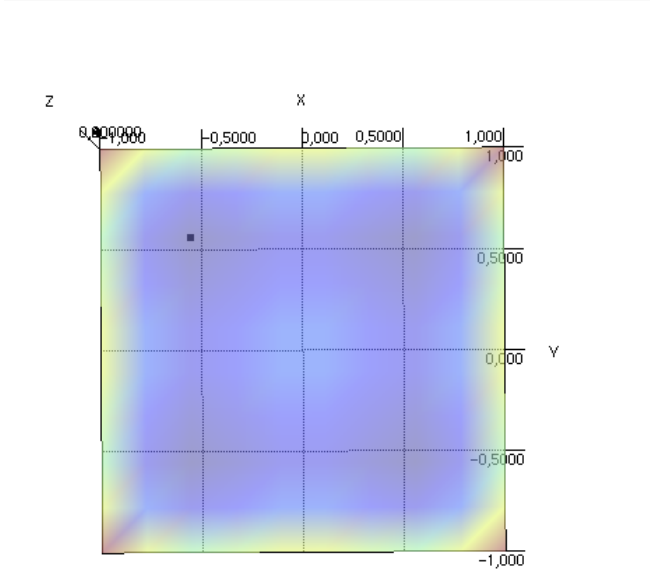
Rysunek 5 prezentuje trójwymiarowy wykres funkcji na przedziale  $x_1, x_2 \in \langle -1, 1 \rangle$  wraz z zaznaczoną najmniejszą obliczoną wartością  $f(x^*) = -0.19219$ ,  $x^*(i) = [\pm 0.55672; \pm 0.55672]$ ,  $i = 1, \dots, 4$ .



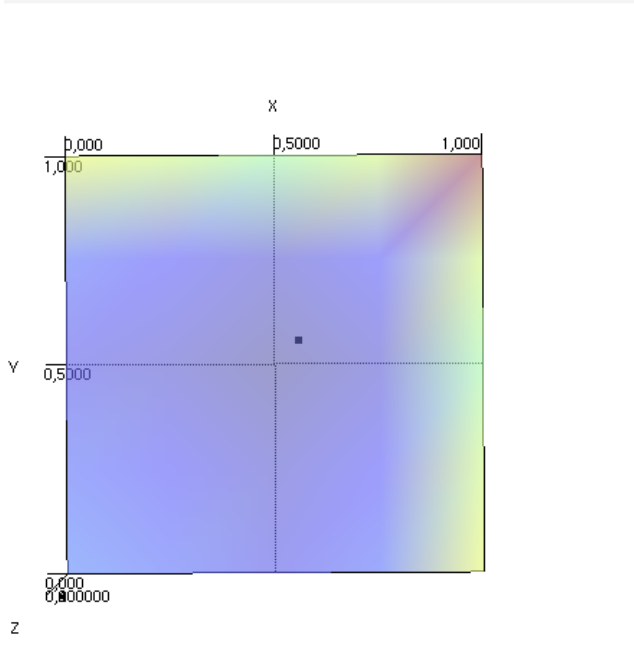
Rysunek 5: Wykres funkcji czterech minimów  $x_1, x_2 \in \langle -1, 1 \rangle$

#### 4.1.1 Wyznaczenie wszystkich minimów

By wyznaczyć wszystkie minima funkcji należało zawęzić przedziały losowania argumentów w taki sposób, by w każdym przedziale było jedno minimum lokalne. Przypadki prezentowane na rysunkach 10, kolejno pokazują każde obliczone minimum zależnie od podanego przedziału. Jak możemy dostrzec na wykresach algorytm znalazł poprawne rozwiązanie w każdym przedziale. Wartości minimum zaznaczone są na wykresach w postaci czarnej kropki oraz wpisane są w najwyższy wiersz tabeli **Rozwiązania** pokazanej po prawej stronie.



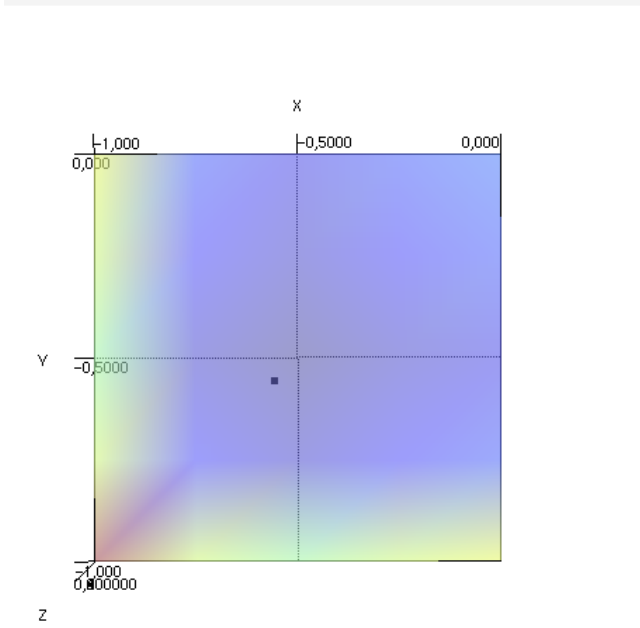
## Rozwiązania

[illegible]

## Rozwiązania

[illegible]

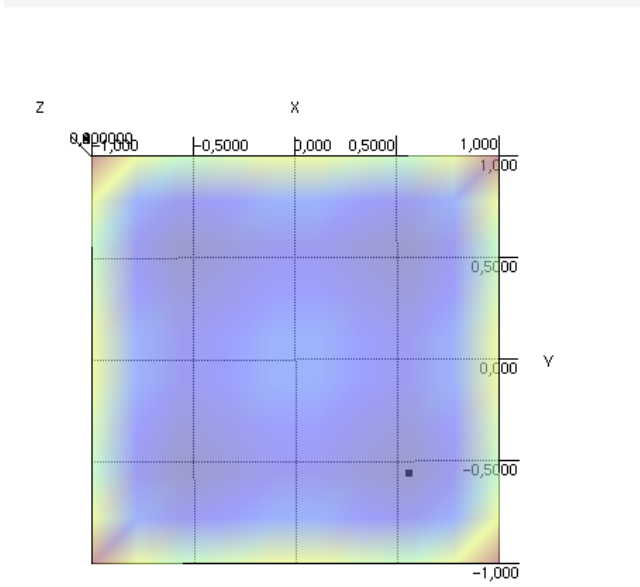
Rysunek 7: Funkcja czterech minimów:  $x_1 \in$



## Rozwiązania

[illegible]

Rysunek 8: Funkcja czterech minimów:  $x_1 \in \langle -1, 0 \rangle, x_2 \in \langle -1, 0 \rangle$



## Rozwiązania

[illegible]

Rysunek 9: Funkcja czterech minimów:  $x_1 \in \langle 0, 1 \rangle, x_2 \in \langle -1, 0 \rangle$

Rysunek 10: Wykres funkcji czterech minimów dla różnych przedziałów

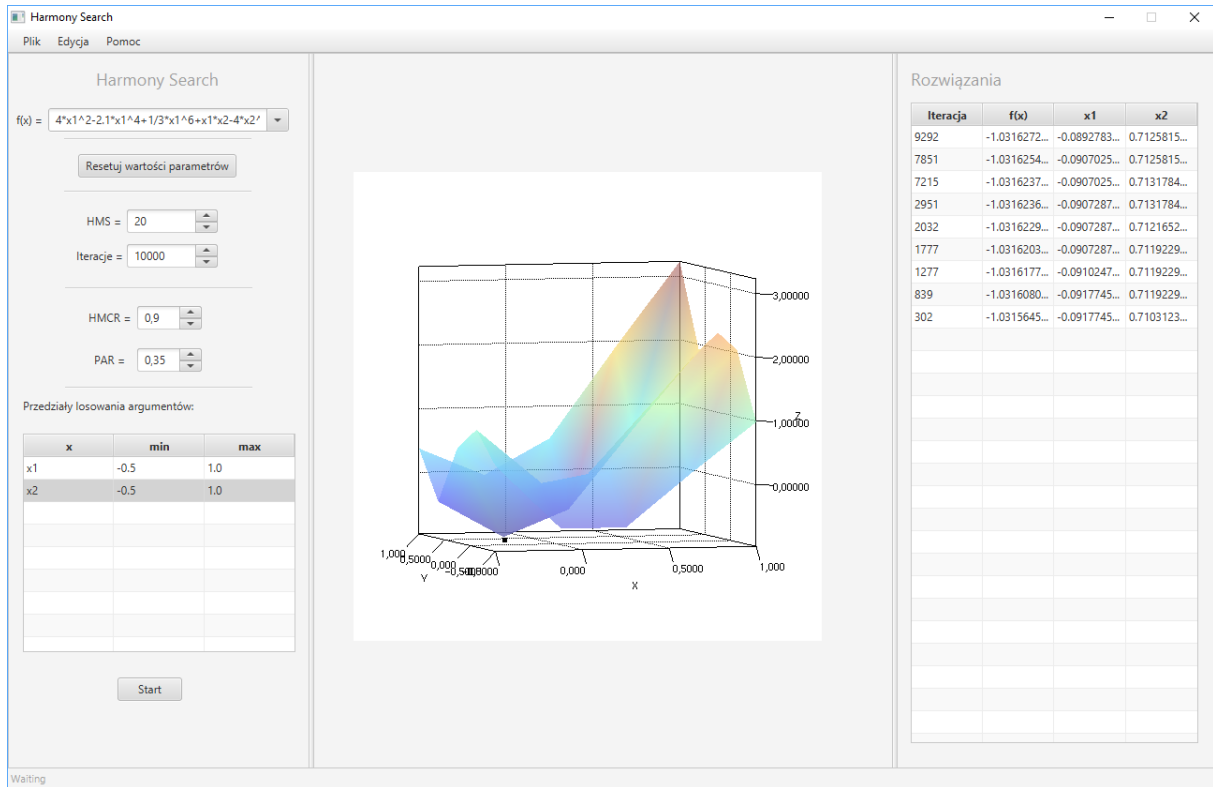


## 4.2 Funkcja testowa Geema

Kolejny przykład przedstawia funkcję opisaną wzorem:

$$f(x_1, x_2) = x_1^4 + x_2^4 - 0.62x_1^2 - 0.62x_2^2.$$

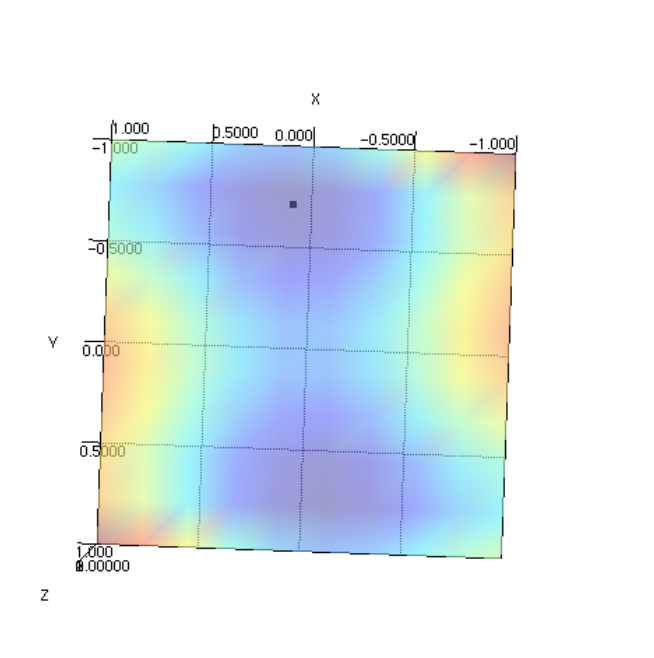
Jest to funkcja zwana szeciogarnym wielbłądem: ma 6 minimów lokalnych, w tym 2 globalne:  $f(x^*) = f(x^{**}) = 1.0316285$ ,  $x^* = [0.08984, -0.71266]$ ,  $x^{**} = [-0.08984, 0.71266]$ . Początkowo zakres dla argumentów funkcji został przyjęty jako  $x_1, x_2 \in \langle -0.5, 1 \rangle$ . Wykres funkcji wraz z rozwiązaniem zaprezentowany jest na rysunku 11.



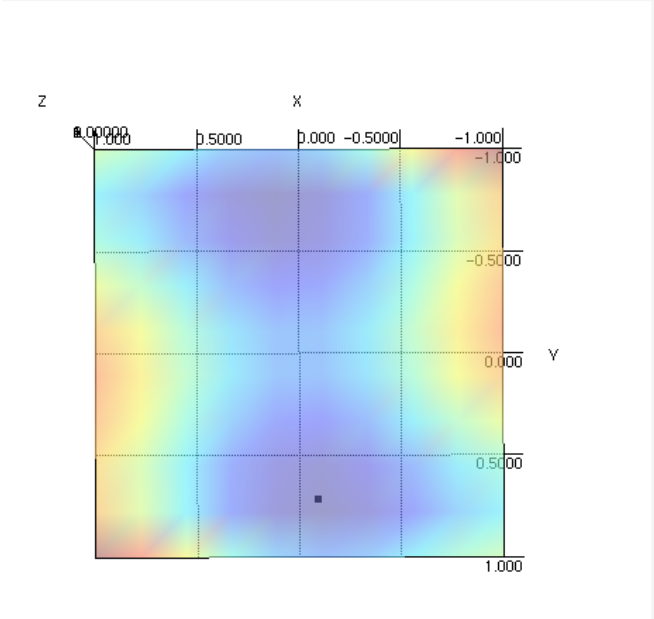
Rysunek 11: Wykres funkcji Geema dla  $x_1, x_2 \in \langle -0.5, 1 \rangle$

### 4.2.1 Zmiana zakresu dla argumentów

Jak w poprzednim przykładzie tak i tutaj zmieniono zakres losowania argumentów na  $x_1, x_2 \in \langle -1, 1 \rangle$ . Zostały znalezione obydwa globalne minima.



## Rozwiązania

[illegible]

## Rozwiązania

[illegible]

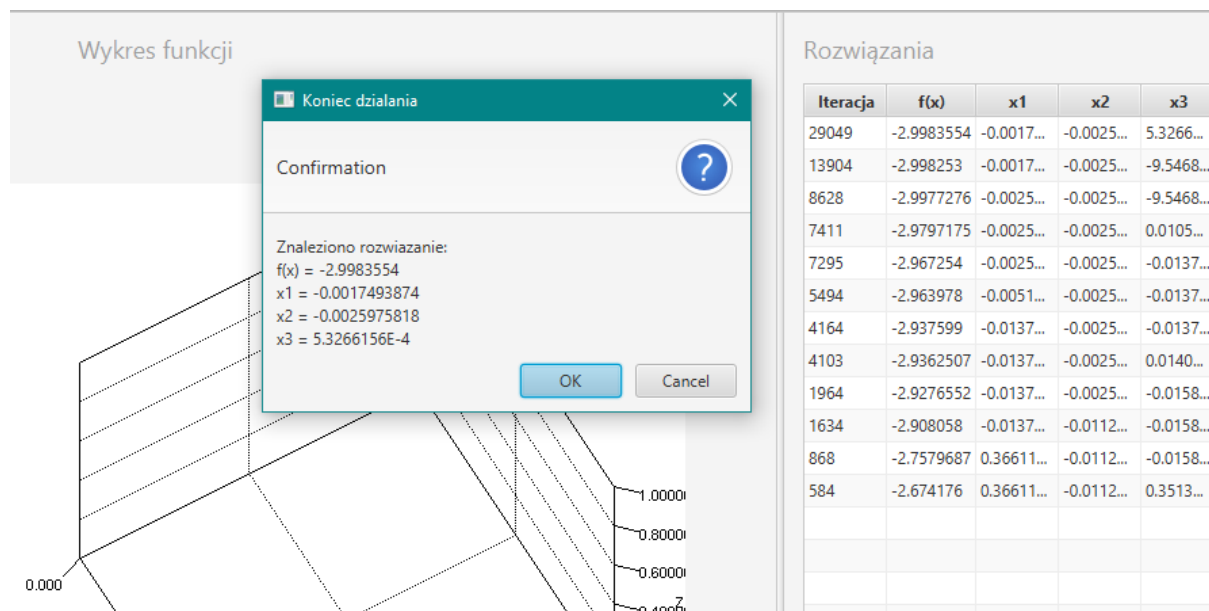
Rysunek 12: Wykresy funkcji Geema dla  $x_1, x_2 \in \langle -1, 1 \rangle$

### 4.3 Funkcja Rastrigina dla $n = 3$

Funkcja Rastrigina jest określona następująco:

$$f(x) = \sum_{i=1}^n x_i^2 - \cos(18x_i), x_i \in (-1, 1).$$

Ma ona jedno globalne minimum  $f(x^*) = -n, x^* = 0$ . W poniższym przypadku przyjęto  $n = 3$ . Program odnajduje je bez większych problemów, chociaż im więcej iteracji, tym bardziej przybliża się do rzeczywistej wartości minimum.



Rysunek 13: Wykres funkcji Rastrigina i znalezione minimum globalne

### 4.4 Funkcja Himmelblau

Funkcja Himmelblau również jak 4.1 posiada cztery minima lokalne zlokalizowane po jednym w każdej ćwiartce na przedziale  $x_{1,2} \in (-5, 5)$ . Funkcja posiada wzór:

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 - 200.$$

Dla pokazania, że funkcja wyszukuje rozwiązania z innego przedziału jak domyślny ograniczono się do przedziału z pierwszej ćwiartki  $x_1, x_2 \in (0, 10)$ . Na wykresie 14 zaprezentowany został wykres funkcji wraz z naniesionym najlepszym rozwiązaniem.

### 4.5 Funkcja Goldsteina-Price'a z czterema minimami lokalnymi

Funkcja Goldsteina-Price'a posiada kilka minimów lokalnych umiejscowionych blisko siebie. Opisuje ją wzór:

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

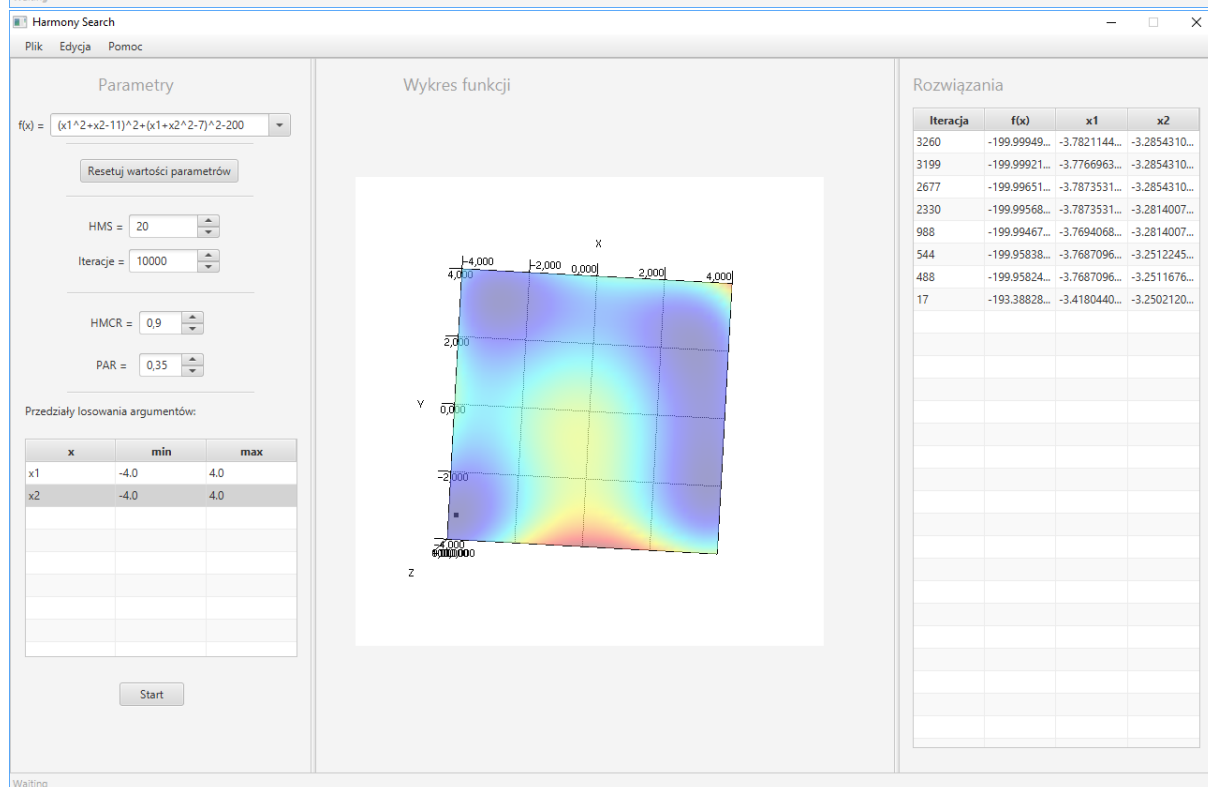
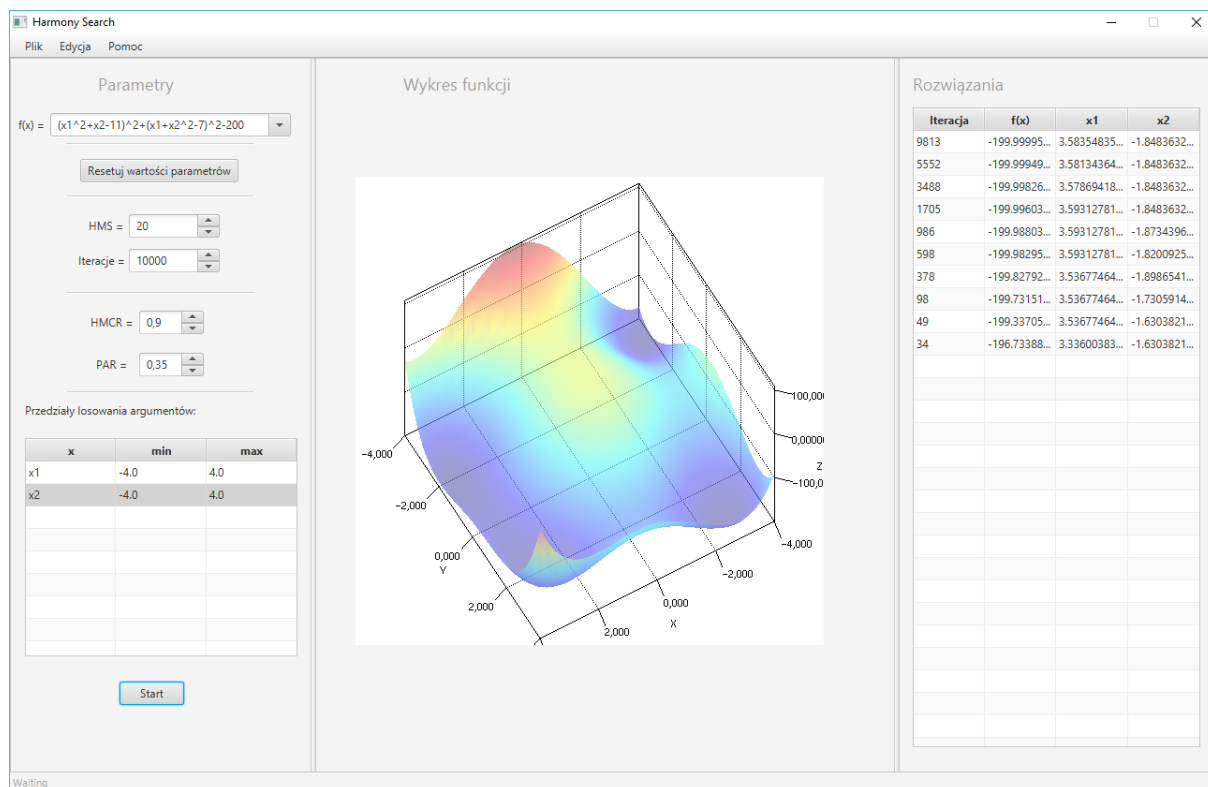
Odpowiednio manewrując parametrami algorytmu, program był w stanie obliczyć wszystkie 4 minima funkcji. Na wykresach 17 oraz 20 pokazano wyznaczone rozwiązania.

### 4.6 Dwuwymiarowa funkcja sinusoidalna

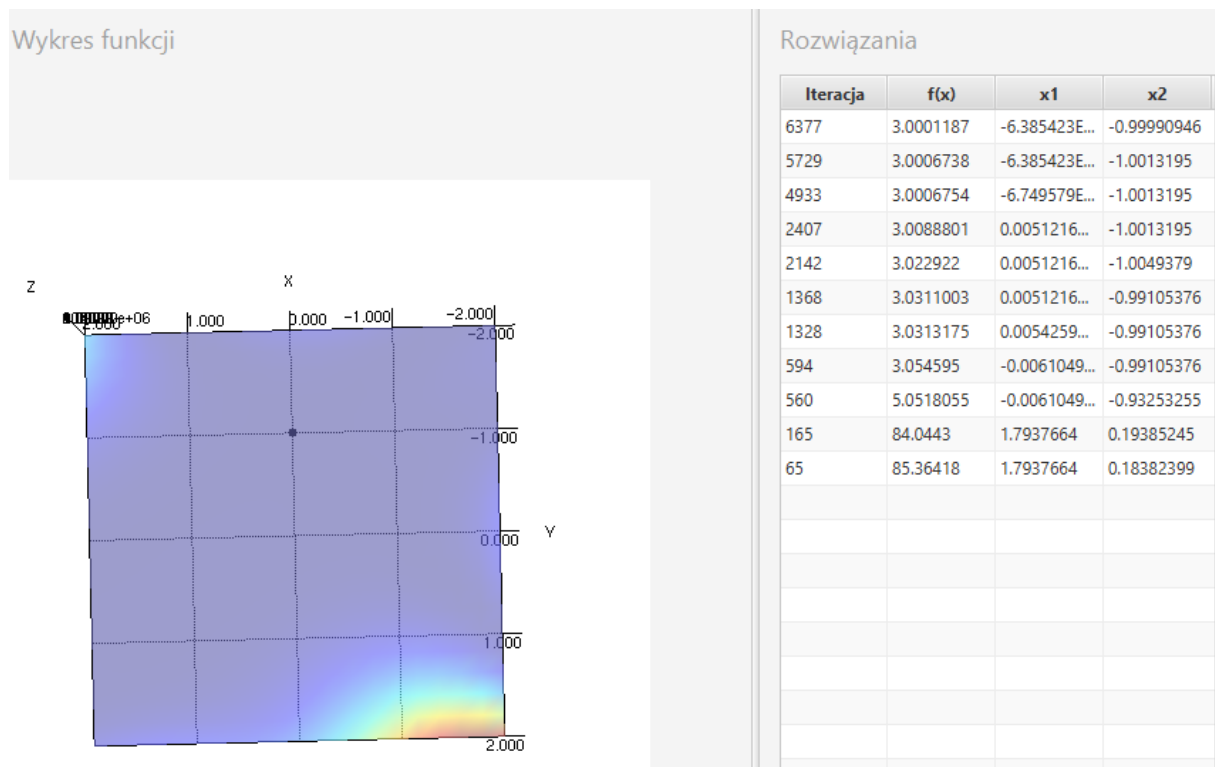
Dwuwymiarowa funkcja sinus posiada bardzo zróżnicowany przebieg oraz wiele równoważnych minimów lokalnych. Wzór funkcji przedstawiony jest równaniem

$$f(x_1, x_2) = \sin x_1 \sin x_2 e^{-(x_1^2 + x_2^2)}.$$

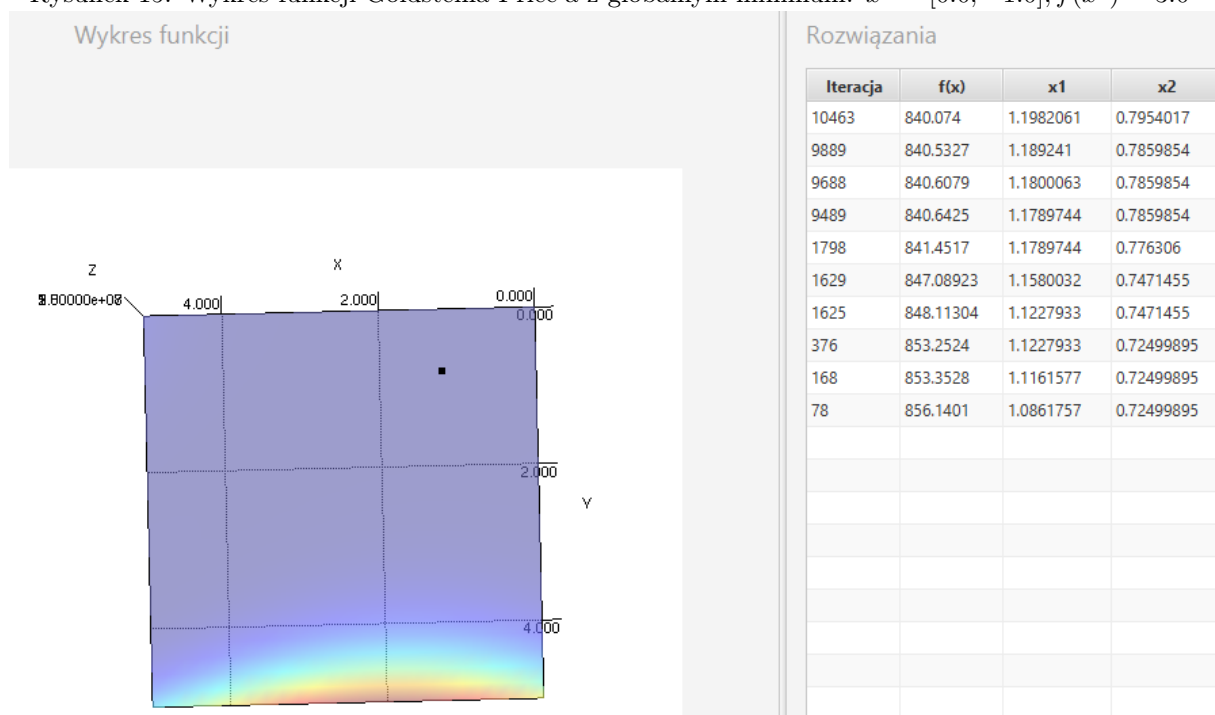
Rysunek 21 prezentuje wykres funkcji dla argumentów z przedziału  $(-1, 1)$ .



Rysunek 14: Wykresy funkcji Himmelblau dla  $x_1, x_2 \in \langle -4, 4 \rangle$

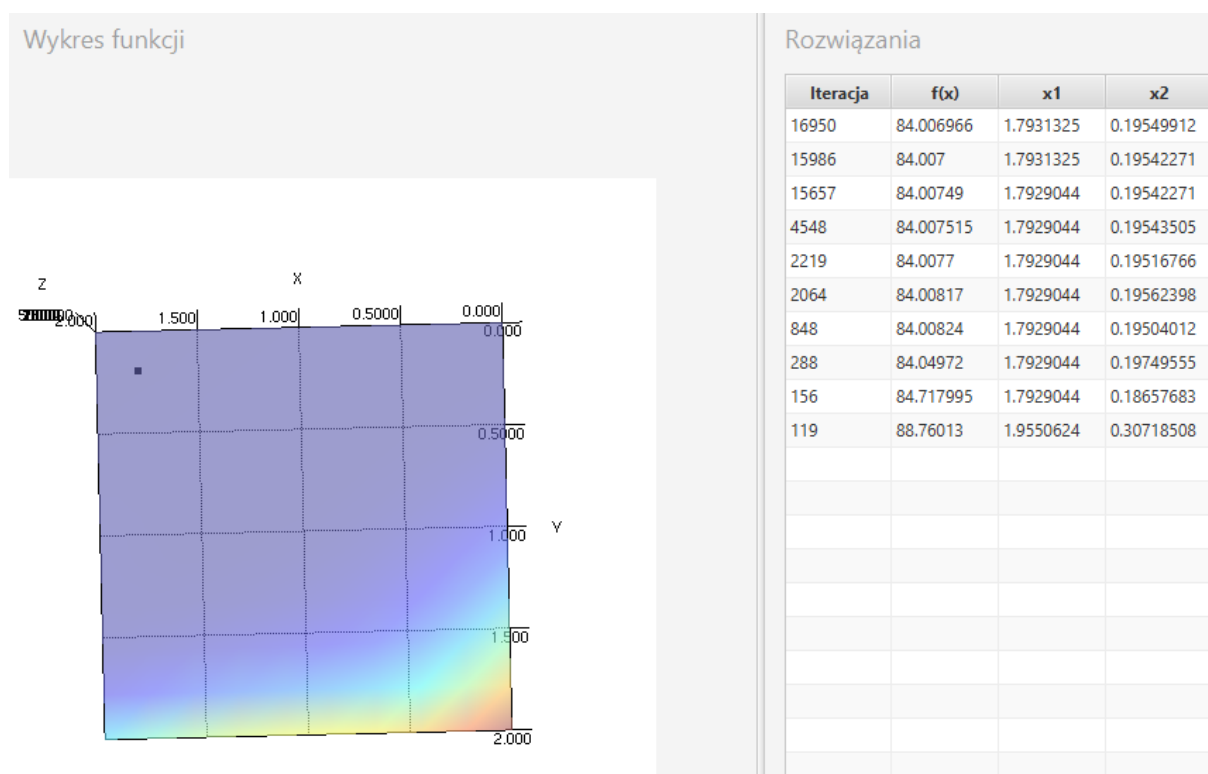


Rysunek 15: Wykres funkcji Goldsteina-Price’a z globalnym minimum:  $x^* = [0.0; -1.0]$ ,  $f(x^*) = 3.0$

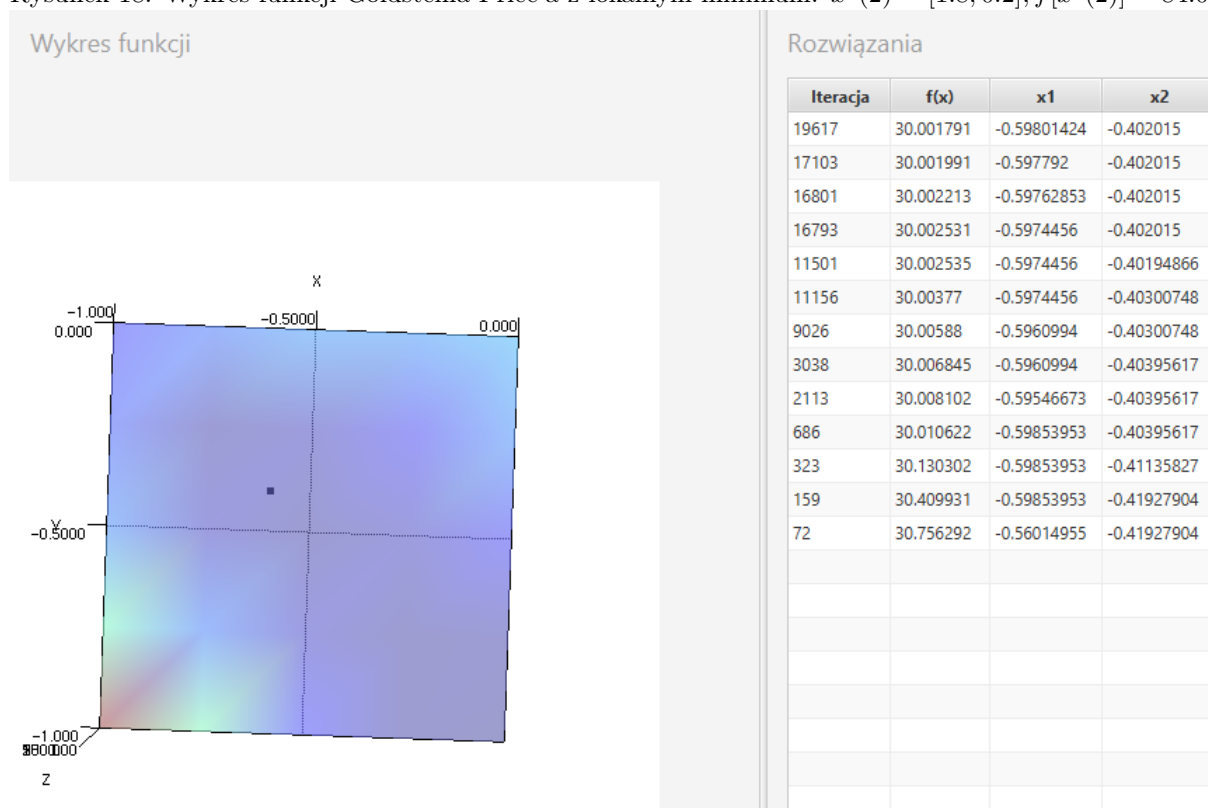


Rysunek 16: Wykres funkcji Goldsteina-Price’a z lokalnym minimum:  $x^*(1) = [1.2; 0.8]$ ,  $f[x^*(1)] = 840.0$

Rysunek 17: Wykresy funkcji Goldsteina-Price’a z lokalnymi minimami

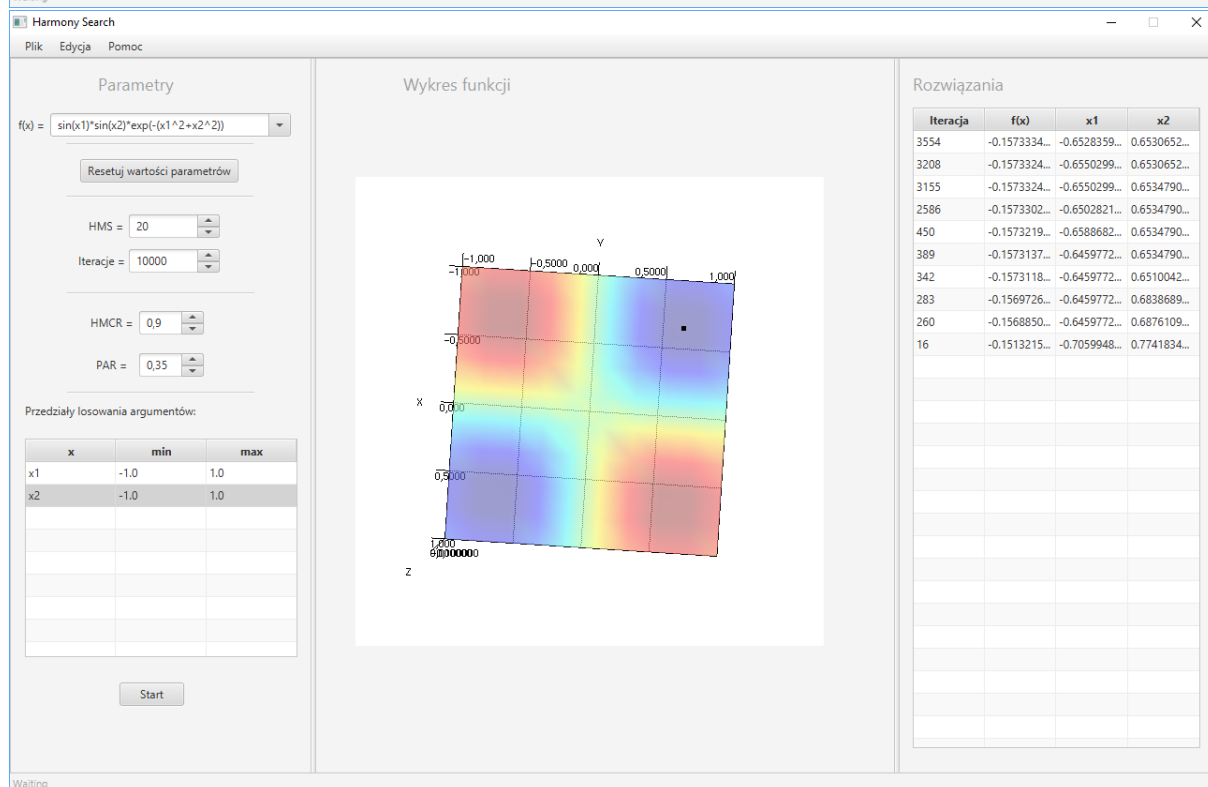
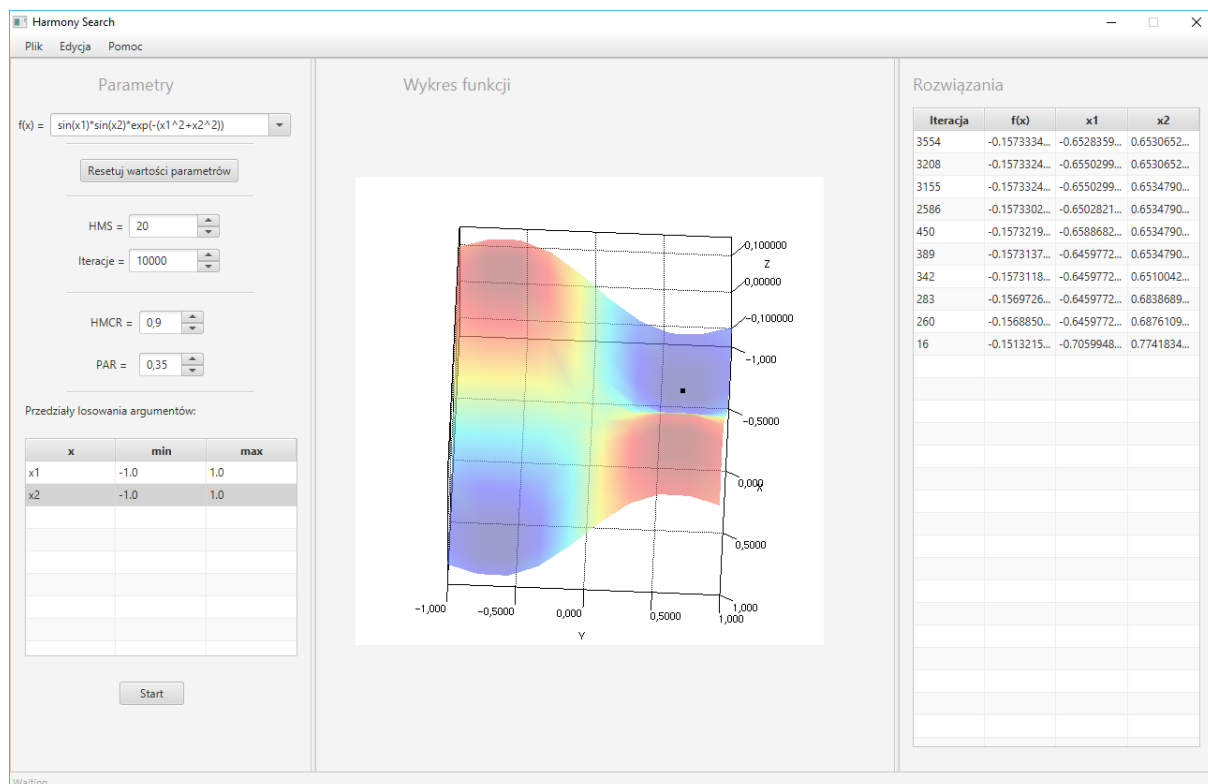


Rysunek 18: Wykres funkcji Goldsteina-Price'a z lokalnym minimum:  $x^*(2) = [1.8; 0.2]$ ,  $f[x^*(2)] = 84.0$



Rysunek 19: Wykres funkcji Goldsteina-Price'a z lokalnym minimum:  $x^*(3) = [-0.6; -0.4]$ ,  $f[x^*(3)] = 30.0$

Rysunek 20: Wykresy funkcji Goldsteina-Price'a z lokalnymi minimami



Rysunek 21: Wykres funkcji sinusoidalnej dla  $x_1, x_2 \in \langle -1, 1 \rangle$

## 4.7 Dwuwymiarowa funkcja sinusoidalna z eksponentą

Ostatnia przedstawiana funkcja ukazuje przypadek połączenia funkcji sinusa-cosinusa z eksponentą. Przypadek jest o tyle ciekawy, że większość wartości funkcji jest dla  $x_1, x_2$  jest do siebie zbliżona. Wzór funkcji przedstawiony jest jako:

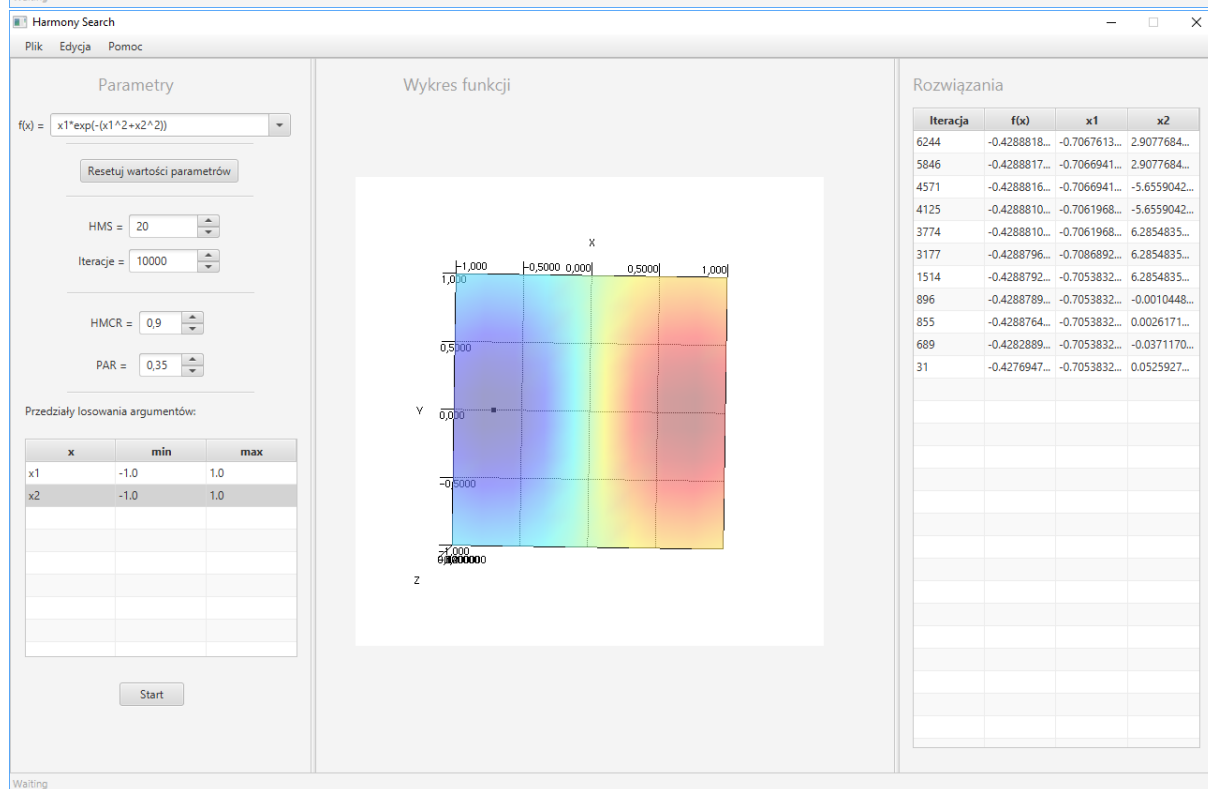
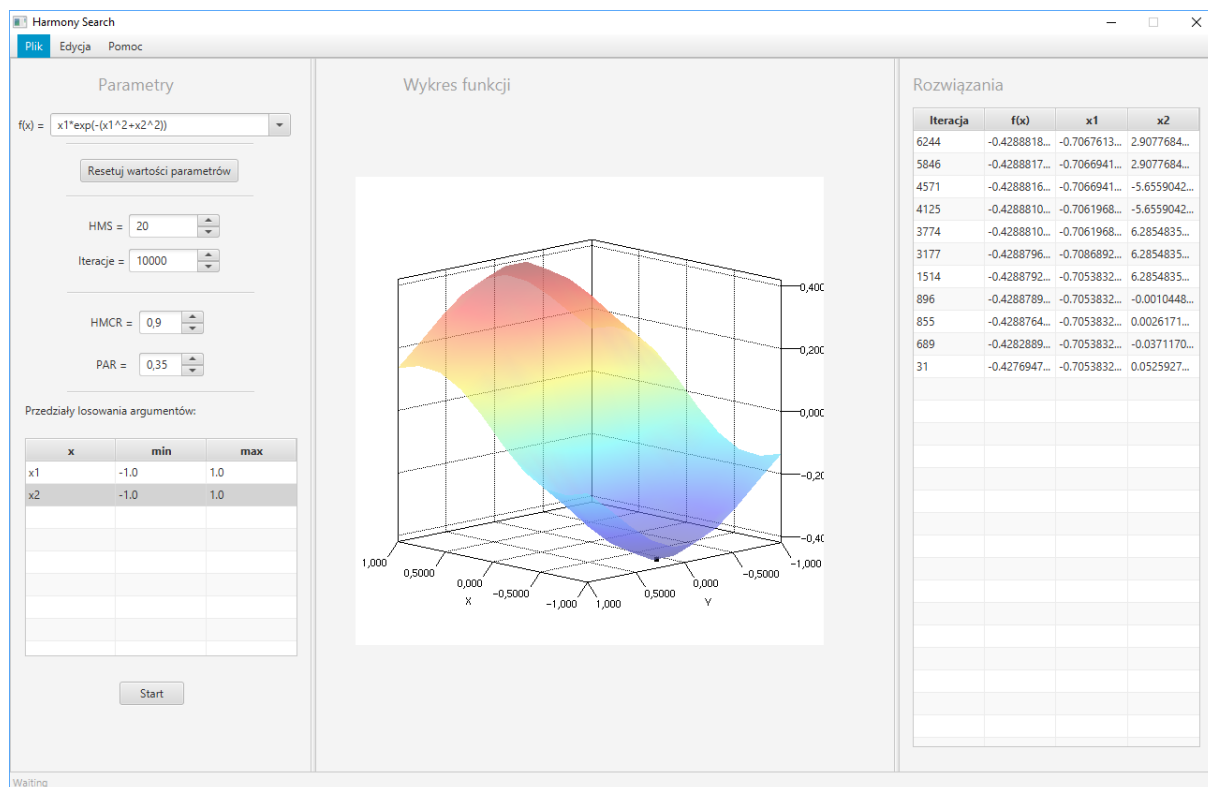
$$f(x) = x_1 e^{-(x_1^2 + x_2^2)}.$$

Rysunek 22 prezentuje działanie programu z obliczoną wartością minimalną. Program poprawnie wyliczył jej wartość, co potwierdza jego odporność na działanie funkcji, dla której większość argumentów jest stała.

## 4.8 Podsumowanie

Sprawdzając działanie programu dla różnych funkcji można stwierdzić, że poprawnie znajduje ich minima lokalne. Algorytm działa na tyle szybko, że dla użytkownika nie odczuwalny jest dyskomfort podczas użytkowania programu. Gdy funkcja posiada więcej niż jedno minimum, można znaleźć wszystkie z nich, odpowiednio kontrolując przedziały losowania argumentów lub liczbę iteracji. Można również sterować prawdopodobieństwami używanymi do generacji kolejnych rozwiązań, co może również pomóc w szybszym lub bardziej dokładnym znalezieniu szukanego minimum danej funkcji.





Rysunek 22: Wykres funkcji sinusoidalnej z eksponentą  $x_1, x_2 \in \langle -1, 1 \rangle$

## 5 Wnioski końcowe

Program obliczający minimum funkcji wielu zmiennych według algorytmu *Harmony Search* działa tak jak założono na początku. Spełniły się domniemania na temat szybkości działania algorytmu. Ponadto szybkość poprawiła się, gdy zamiast zwykłej tablicy dla HM przyjęto *SortedSet*. Program najdokładniej przybliży rozwiązanie, gdy zakres wartości  $x$  będzie jak najbliższy minimum funkcji. Dodatkowo, gdy ograniczymy ilość iteracji, program zakończy szybciej swoje działanie, lecz kosztem znalezienia gorszego rozwiązania. Gdy wymagane jest obliczenie każdego z minimum funkcji możemy je znaleźć i obliczyć jego wartość, o ile znamy przedział, w którym jest ono umiejscowione. Poprzez odpowiednie ustawienie parametrów algorytmu i przedziałów losowania argumentów, jesteśmy w stanie wskazać kierunek poszukiwań lokalnych optimum funkcji.

Sam algorytm Harmony Search zyskuje coraz większą popularność w praktycznych zastosowaniach [4]. Używa się do znajdowania rozwiązań problemów VRP (*ang. vehicle routing problem*, przykładowo szukanie optymalnej trasy szkolnych autobusów), projektowaniu sieci wodociągowych, systemów zapór wodnych czy pompy ciepła w satelitach. Zastosowanie znajduje także w innych dziedzinach: przewidywaniu struktury RNA, obróbce medycznych obrazów i onkologii, robotyce, śledzeniu wizualnym i wielu innych. Bardzo często daje lepsze rezultaty przy mniejszych błędach niż stosowane wcześniej metody, nie wymaga żadnych informacji o gradiencie funkcji, a także możliwe jest stosowanie do problemów dyskretnych. Wadą jest natomiast konieczność określenia parametrów działania, typowych dla algorytmów metaheurystycznych.

## Literatura

- [1] Kang Seok Lee, Zong Woo Geem, and G.V. Loganathan. *A New Heuristic Optimization Algorithm: Harmony Search*. 2004.
- [2] Andrzej Kowal. *Algorytm Harmony Search*.
- [3] Mariusz Gromada. *Dokumentacja biblioteki mXParser*. dostęp 27.05.2018. "<http://mathparser.org/>".
- [4] Zong Woo Geem. *Music-Inspired Optimization Algorithm: Harmony Search*. 2004.